Package ‘ggVennDiagram’

July 7, 2021

Type Package
Title A ‘ggplot2’ Implement of Venn Diagram
Version 1.1.4
Maintainer Chun-Hui Gao <gaospecial@gmail.com>
Description Easy-to-use functions to generate 2-7 sets Venn plot in publication quality.

‘ggVennDiagram’ plot Venn using well-defined geometry dataset and ‘ggplot2’. The shapes of 2-4 sets
Venn use circles and ellipses, while the shapes of 4-7 sets Venn use irregular polygons (4 has both forms), which
are developed and imported from another package ‘venn’, authored by Adrian Dusa. We pro-
vided internal functions to
integrate shape data with user provided sets data, and calculated the geometry of every re-
gions/intersections
of them, then separately plot Venn in three components: set edges, set labels, and regions.
From version 1.0, it is possible to customize these components as you demand in ordinary ‘gg-
plot2’ grammar.
Depends R (>= 3.5.0)
Imports sf, ggplot2, dplyr, stringr, magrittr, methods, purrr, tibble,
plotly, RVenn
URL https://github.com/gaospecial/ggVennDiagram
License GPL-3
Encoding UTF-8
RoxygenNote 7.1.1
Suggests testthat (>= 2.1.0), knitr, rmarkdown, tidyr, venn,
RCOLORBrewer
VignetteBuilder knitr
NeedsCompilation no
Author Chun-Hui Gao [aut, cre] (<https://orcid.org/0000-0002-1445-7939>),
Guangchuang Yu [ctb] (<https://orcid.org/0000-0002-6485-8781>),
Adrian Dusa [ctb]
Repository CRAN
Date/Publication 2021-07-07 10:50:02 UTC
Description

Venn diagram is frequently used in scientific studies of many fields. This package incorporates state-of-art Venn plot tools and provides a set of easy-to-use functions to plot Venn. By dealing with a user-provided list, which contains the sets of Venn, ‘ggVennDiagram’ returns a structured data that can be used to plot Venn. The data contains three slots: 1) the edge of Venn sets; 2) the separated regions of Venn sets; 3) the labels of Venn sets. By help from the package ‘venn’, it is possible to draw Venn diagram up to 7 sets.
build_shape

**Helper function to add shape**

**Description**

Helper function to add shape

**Usage**

```r
build_shape(
  edge,
  label,
  nsets = length(edge),
  shape_id,
  type = c("ellipse", "triangle", "polygon", "circle")
)
```

**Arguments**

- `edge` a list of xy matrix
- `label` a list of xy matrix
- `nsets` 2:7
- `shape_id` a unique id
- `type` c("ellipse","triangle","polygon","circle")

**Value**

a tibble with columns: nsets, type, shape_id, component, id, xy.

circle

**generating a circle**

**Description**

generating a circle

**Usage**

```r
circle(x = 0, y = 0, r = 1, n = 100)
```

**Arguments**

- `x`, `y` center of circle
- `r` radius of circle
- `n` number of points for polygon object (resolution)
Value

   a matrix representing circle coordinates

Examples

   # plot the default circle
   library(ggVennDiagram)
   library(sf)
   circle() %>% st_linestring() %>% plot()

combinations

   all possible combinations of n sets

Description

   all possible combinations of n sets

Usage

   combinations(n)

Arguments

   n        dim

discern, Polygon-method

   calculate the difference region of 'Polygon' object

Description

   calculate the difference region of 'Polygon' object

Usage

   ## S4 method for signature 'Polygon'
   discern(venn, slice1, slice2 = "all")

Arguments

   venn        Venn/Polygon object
   slice1  first slice of Venn object
   slice2  second slice of Venn object, default is all except the first slice
calculate region of Venn

calculate the unique region defined by 'Venn' object and the parameter 'slice'

Usage

discern_overlap(venn, slice = "all")

## S4 method for signature 'Venn'
discern_overlap(venn, slice = "all")

## S4 method for signature 'Polygon'
discern_overlap(venn, slice = "all")

Arguments

venn Venn object
slice a numeric vector indicating the index of slice, default is "all"

Value

region items

Examples

library(ggVennDiagram)
venn <- Venn(list(A=1:3,B=2:5,C=c(1L,3L,5L)))

discern_overlap(venn, slice = "all")
# is equal to
overlap(venn, slice = "all")

# however, `discern_overlap()` only contains specific region
discern_overlap(venn, slice = 1:2)
**Description**

This function is derived from `VennDiagram::ell2poly`, we modified it and then it can generating a closed ellipse, which is a requirement for further transformation to a POLYGON sf object.

**Usage**

```
ellipse(x = 0, y = 0, a = 2, b = 1, rotation = 0, n = 100)
```

**Arguments**

- `x, y`: the coordinates of ellipse center
- `a`: radius of short arm
- `b`: radius of long arm
- `rotation`: rotation in degree
- `n`: number of points

**Value**

A matrix representing ellipse coordinates

**Examples**

```
# plot the default ellipse
library(sf)
library(ggVennDiagram)
ellipse() %>% st_linestring() %>% plot()
```

---

**fancy_2d_circle**

two dimension circle

**Description**

Two dimension circle

**Usage**

```
fancy_2d_circle(parameters = NULL, n = 100)
```

**Arguments**

- `parameters`: will pass to shape generators
- `n`: count of points to shape this polygon
fancy_3d_circle

Description
fancy 3d circle

Usage
fancy_3d_circle(parameters = NULL, n = 100)

Arguments
- parameters: will pass to shape generators
- n: count of points to shape this polygon

fancy_4d_ellipse

Description
fancy 4d ellipse from 'VennDiagram'

Usage
fancy_4d_ellipse(parameters = NULL, n = 100)

Arguments
- parameters: will pass to shape generators
- n: count of points to shape this polygon

Value
a list of coordinates matrix
**fancy_4d_ellipse_label**

*helper function to set label position*

---

**Description**

helper function to set label position

**Usage**

```r
fancy_4d_ellipse_label(position = NULL)
fancy_3d_circle_label(position = NULL)
fancy_2d_circle_label(position = NULL)
fancy_6d_triangle_label(position = NULL)
label_position(position)
```

**Arguments**

- `position`: a data.frame containing label coordinates

**Details**

- `label_position`: basal wrapper for label position
- `fancy_6d_triangle_label`: 6 sets triangle label position work with `fancy_6d_triangle`
- `fancy_4d_ellipse_label`: 4 sets ellipse label position work with `fancy_4d_ellipse`
- `fancy_3d_circle_label`: 3 sets circle label position work with `fancy_3d_circle`
- `fancy_2d_circle_label`: 2 sets circle label position work with `fancy_2d_circle`

**Value**

a list of matrix

**Examples**

```r
fancy_4d_ellipse_label()
fancy_2d_circle_label()
```
fancy_6d_triangle

---

fancy_6d_triangle  Six dimension triangle

**Description**

Six dimension triangle

**Usage**

fancy_6d_triangle(parameters = NULL)

**Arguments**

- **parameters**  will pass to shape generators

---

get_shape_data  get applicable shape data for Venn object

**Description**

ggVennDiagram stores shapes as internal data. You may see all the shapes by using `plot_shapes()`.

**Usage**

get_shape_data(nsets, ...)

**Arguments**

- **nsets**  number of sets
- **...**  Arguments passed on to process_data

- **venn**  a Venn object

**Value**

a tibble describing specific shape

**Examples**

get_shape_data(nsets = 3, type == "polygon")
ggVennDiagram

Description

ggVennDiagram main parser

Usage

ggVennDiagram(
  x,
  category.names = names(x),
  show_intersect = FALSE,
  set_color = "black",
  set_size = NA,
  label = c("both", "count", "percent", "none"),
  label_alpha = 0.5,
  label_geom = c("label", "text"),
  label_color = "black",
  label_size = NA,
  label_percent_digit = 0,
  label_txtWidth = 40,
  edge_lty = "solid",
  edge_size = 1,
  ...
)

Arguments

  x  list of items
  category.names  default is names(x)
  show_intersect  if TRUE the text can be visualized by 'plotly'
  set_color  color of set labels ("black")
  set_size  size of set labels (NA)
  label  format of region labels, select one from c("count","percent","both","none")
  label_alpha  set 0 to remove the background of region labels
  label_geom  layer of region labels, choose from c("label", "text")
  label_color  color of region labels ("black")
  label_size  size of region labels (NA)
  label_percent_digit  number of digits when formatting percent label (0)
  label_txtWidth  width of text used in showing intersect members, will be ignored unless show_intersection is TRUE (40)
edge_lty  line type of set edges ("solid")
edge_size line width of set edges (1)
... Other arguments passed on to downstream functions.

Value
A ggplot object

Examples
library(ggVennDiagram)
ggVennDiagram(x)  # 4d venn
ggVennDiagram(x[1:3])  # 3d venn
ggVennDiagram(x[1:2])  # 2d venn

Description
calculate the overlap region of ‘Polygon’ object

Usage
## S4 method for signature 'Polygon'
overlap(venn, slice = "all")

Arguments

venn  Venn object
slice  a numeric vector indicating the index of slice, default is "all"

Description
join the shape data with set data

Usage
plotData_add_venn(plotData, venn)
Arguments

plotData  a VennPlot object that stores plot shapes
venn  a Venn object that stores set values

plot_shapes  *plot all shapes provided by internal dataset*

Description

These shapes are mainly collected from the package ‘venn’, and ‘VennDiagram’. For Venn plot with more than 4 sets, it is usually impossible to plot with simple circle or ellipse. So we need to use a predefined coordinates in plot.

Usage

plot_shapes()

Details

- Shape 101, 201, 301, 401, 402, 501, 502, 601 and 701 are from ‘venn’
- Shape 401f is from ‘VennDiagram’
  see ‘data-raw/shapes.R’ to find how we incorporate these data.

Examples

plot_shapes()

plot_venn  *plot codes*

Description

plot codes

Usage

plot_venn(
  x,
  show_intersect,
  set_color,
  set_size,
  label,
  label_geom,
  label_alpha,
  label_color,
Polygon

Arguments

<table>
<thead>
<tr>
<th>x</th>
<th>list of items</th>
</tr>
</thead>
<tbody>
<tr>
<td>show_intersect</td>
<td>if TRUE the text can be visualized by 'plotly'</td>
</tr>
<tr>
<td>set_color</td>
<td>color of set labels (&quot;black&quot;)</td>
</tr>
<tr>
<td>set_size</td>
<td>size of set labels (NA)</td>
</tr>
<tr>
<td>label</td>
<td>format of region labels, select one from c(&quot;count&quot;,&quot;percent&quot;,&quot;both&quot;,&quot;none&quot;)</td>
</tr>
<tr>
<td>label_geom</td>
<td>layer of region labels, choose from c(&quot;label&quot;,&quot;text&quot;)</td>
</tr>
<tr>
<td>label_alpha</td>
<td>set 0 to remove the background of region labels</td>
</tr>
<tr>
<td>label_color</td>
<td>color of region labels (&quot;black&quot;)</td>
</tr>
<tr>
<td>label_size</td>
<td>size of region labels (NA)</td>
</tr>
<tr>
<td>label_percent_digit</td>
<td>number of digits when formatting percent label (0)</td>
</tr>
<tr>
<td>label_txtWidth</td>
<td>width of text used in showing intersect members, will be ignored unless show_intersection is TRUE (40)</td>
</tr>
<tr>
<td>edge_lty</td>
<td>line type of set edges (&quot;solid&quot;)</td>
</tr>
<tr>
<td>edge_size</td>
<td>line width of set edges (1)</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments passed on to downstream functions.</td>
</tr>
</tbody>
</table>

Value

ggplot object, or plotly object if show_intersect is TRUE

Description

Polygon constructor

Usage

Polygon(sets)

### S4 method for signature 'ANY'

Polygon(sets)
Arguments
sets a list containing multiple simple features

Polygon-class An S4 class to represent multiple polygons.

Description
An S4 class to represent multiple polygons.

Slots
sets A list contains sets
names The names of the ‘sets’ if has names. If the ‘list’ doesn’t have names, the sets will be named as “Set_1”, “Set_2” and so on.

process_data get plot data

Description
get plot data

Usage
process_data(venn, ...)

## S4 method for signature 'Venn'
process_data(venn, ...)

Arguments
venn a Venn object
... apply filter to internal shapes. i.e. shape_id == "601", type == "polygon"

Examples
## Not run:
venn <- Venn(list(A=1:3,B=2:5,C=4:8))
data <- process_data(venn)

## End(Not run)
shape_generator

functions to generate ellipse, circle, triangle and other shapes, which will be used in Venn plot

Description

defined a triangle by three points

Usage

triangle(xy = c(0, 0, 1, 0, 0, 1))

Arguments

xy coordinates of the three points defining a triangle

Value

a matrix with xy coordinates

Examples

# triangle coordinates
library(ggVennDiagram)
library(sf)
triangle()

# plot a new triangle
triangle(c(-1,0,1,0,0,2)) %>% st_linestring() %>% plot()
Venn-class  

An S4 class to represent multiple sets.

Description

This class is adopted from ‘RVenn’. Since ‘RVenn’ doesn’t export this class, I have to copy its codes hereafter to use it.

Slots

sets A list object containing vectors in the same type.

names The names of the sets if it has names. If the list doesn’t have names, the sets will be named as "Set_1", "Set_2", "Set_3" and so on.

vennplot-shapes  

shapes: internal shape data

Description

A collection of geometric shapes, which defined the edge and label of sets in a Venn plot. Use plot_shapes() to see some of them.

Usage

shapes

Format

A tibble with 6 columns

- nsets: number of sets, from 1-7.
- type: ellipse, circle or triangle
- shape_id: to separate different shapes
- component: each shape has two components, ‘setEdge’ and ‘setLabel’
- id: to separate edges/labels of a shape. For example, 4 sets shape will have ids of 1-4.
- xy: coordinates

Source

- venn::sets
- library(VennDiagram)
- Wiki
VennPlotData constructor

VennPlotData

Description

VennPlotData constructor

Usage

VennPlotData(setEdge, setLabel)

## S4 method for signature 'ANY'
VennPlotData(setEdge, setLabel)

Arguments

setEdge a list of coordinates matrix defining Venn set edges
setLabel a list of coordinates matrix defining Venn set labels#

Value

a S4 class VennPlotData object

VennPlotData-class

An S4 class to represent Venn plot components.

Description

An S4 class to represent Venn plot components.

Slots

setEdge a list of coordinates matrix defining Venn set edges
setLabel a list of coordinates matrix defining Venn set labels
region the feature region will be calculated automatically with 'setEdge'
venn_data

Get VennPlotData slot

Description
Get VennPlotData slot
Prepare Venn data

Usage
venn_region(obj)
venn_setedge(obj)
venn_setlabel(obj)
process_setEdge_data(venn)
process_setLabel_data(venn)
process_region_data(venn)

Arguments
obj a S4 class ‘VennPlotData’ object
venn a Venn object

Value
a tibble, ‘sf’ object
a tibble

Examples
## Not run:
# obj is VennPlotData
venn_region(obj) # return region data
venn_setlabel(obj) # return setLabel data
venn_setedge(obj) # return setEdge data

## End(Not run)
x <- list(
  A = sample(letters, 8),
  B = sample(letters, 8),
  C = sample(letters, 8),
  D = sample(letters, 8)
)
venn_data

venn <- Venn(x)
process_region_data(venn)
process_setEdge_data(venn)
process_setLabel_data(venn)
Index

* datasets
  - vennplot-shapes, 16

build_shape, 3

circle, 3
combinations, 4

discern, Polygon-method, 4
discern_overlap, 5
discern_overlap, Polygon-method
  (discern_overlap), 5
discern_overlap, Venn-method
  (discern_overlap), 5

ellipse, 6

fancy_2d_circle, 6
fancy_2d_circle_label
  (fancy_4d_ellipse_label), 8
fancy_3d_circle, 7
fancy_3d_circle_label
  (fancy_4d_ellipse_label), 8
fancy_4d_ellipse, 7
fancy_4d_ellipse_label, 8
fancy_6d_triangle, 9
fancy_6d_triangle_label
  (fancy_4d_ellipse_label), 8

get_shape_data, 9
ggVennDiagram, 10
ggVennDiagram-package, 2

label_position
  (fancy_4d_ellipse_label), 8

overlap, Polygon-method, 11

plot_shapes, 12
plot_venn, 12
plotData_add_venn, 11

Polygon, 13
Polygon, ANY-method (Polygon), 13
Polygon-class, 14
process_data, 9, 14
process_data, Venn-method
  (process_data), 14
process_region_data (venn_data), 18
process_setEdge_data (venn_data), 18
process_setLabel_data (venn_data), 18

shape_generator, 15
shapes (vennplot-shapes), 16

triangle, 15

Venn-class, 16
venn_data, 18
venn_region (venn_data), 18
venn_setedge (venn_data), 18
venn_setlabel (venn_data), 18
vennplot-shapes, 16
VennPlotData, 17
VennPlotData, ANY-method (VennPlotData), 17
VennPlotData-class, 17